REMARKS

With respect to the provisional obviousness-type double patenting rejection, it is again requested that this rejection be deferred until such time as one of the two cases at issue is indicated to be allowable. Until that time, it is difficult, if not impossible, to ascertain the true extent of any double patenting issue. Only when the allowed claims in one case can be compared to the pending claims in another case can it truly be determined whether or not a double patenting issue arises.

With respect to the rejection of the claims, reconsideration is requested because each of the claims requires granting the resource to the thread of instructions.

In the cited reference to Wang, what is at issue is a variable. The problem is that each of a plurality of threads may try to initialize the variable at the same time. As explained at the bottom of column 1, if a first thread executes the initialization of static variable A, and before the initialization is complete, the control of the CPU is given to a second thread, the second thread can try to reference static variable A. Since the initialization of A has not been completed by the first thread, a run time error can occur.

As explained in column 3, lines 28-35, "the present invention provides a means to eliminate this problem" where "this problem" is that in a multithreaded application, a static variable can be initialized more than once. Thus, there is no need to grant the variable to a thread after it has been initialized. It is not a resource that can only be used by one thread at a time. It is simply a variable that should not be used by any thread until any one thread initializes it. Thus, once it has been initialized, all the threads can use it.

Therefore, there is no teaching, nor any reason, to grant the resource to the thread of instructions after changing the thread to the active state.

Reconsideration of claim 1 is requested on this basis.

Reconsideration of the rejection of claim 11 on the same basis is requested.

Likewise, claim 15 calls for an execution unit to grant the resource to the thread of instructions when the resource becomes available. Here, in the cited reference, the resource never really becomes available and it is never granted to any thread after it is initialized. The threads are just prevented from using it until such time as it is initialized and then there is nothing to stop them all from using it.

Therefore, reconsideration is respectfully requested.

Respectfully submitted,

Date: December 10, 2008

/Timothy N. Trop/
Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
713/468-8880 [Phone]
713/468-8883 [Fax]

Attorneys for Intel Corporation